Theoretical Computer Science 99 (1992) 279–290
Elsevier

Logarithmic advice classes*

José L. Balcázar

Department of Software (Llenguatges i Sistemes Informàtics), Universitat Politecnica de Cataluna, Pau Gargallo 5, E-08028 Barcelona, Spain

Uwe Schöning

Universität Ulm, Abt. Theoretische Informatik, Postfach 4066, D-7900 Ulm, Germany

Communicated by P. Young
Received November 1988
Revised February 1990

Abstract

Balcázar, J.L. and U. Schöning, Logarithmic advice classes, Theoretical Computer Science 99 (1992) 279–290.

Karp and Lipton (1980) introduced the notion of nonuniform complexity classes where a certain amount of additional information, the advice, is given for free. The advice only depends on the length of the input. Karp and Lipton initiated the study of classes with either logarithmic or polynomial advice; however, later Yap (1983), Schöning (1984), Balcázar et al. (1987) and Ko and Schöning (1985) concentrated on the study of classes of the form $\mathcal{C}/poly$, where $\mathcal{C}$ is P, NP, or PSPACE, and *poly* denotes a polynomial-size advice.

This paper considers classes of the form $\mathcal{C}/log$. As a main result it is shown that in the context of an NP/*log* computation, log-bounded advice is equivalent to a sparse oracle in **NP**. In contrast, it has been shown that a poly-bounded advice corresponds to an *arbitrary* sparse oracle set.

Furthermore, a general theorem is presented that generalizes Karp and Lipton's "round-robin tournament" method.

1. Preliminaries

In this section we introduce the relevant notation and review some important results. For more detailed information we refer the reader to [3, 20].

All sets considered are languages over some alphabet Σ, $|\Sigma|>1$. For a string $x\in\Sigma^*$, $|x|$ denotes its length. For a set A and an integer n, A_n denotes the set $\{x\in A \mid |x|=n\}$. We call a set S *sparse* if there is a polynomial p such that for all n, the cardinality of A_n is at most $p(n)$.

* This work was supported by the Deutsche Forschungsgemeinschaft, grant No. Scho 302/2, and by a grant of CIRIT (Generalitat de Catalunya).

The classes P and NP have their standard definitions. We also refer to their relativized versions $P(A)$, $NP(A)$ and $P(\mathscr{C})$, $NP(\mathscr{C})$, where A is a set and $\mathscr{C}$ is a class of sets. For example, $P(\mathscr{C})$ is the class of sets that can be recognized by polynomial-time bounded deterministic oracle machines using oracle sets in $\mathscr{C}$.

The *polynomial-time hierarchy* [23] consists of the classes Δ_i, Σ_i, Π_i for $i \geqslant 0$, and is defined as

$$\Delta_0 = \Sigma_0 = \Pi_0 = P,$$

and

$$\Delta_{i+1} = P(\Sigma_i), \qquad \Sigma_{i+1} = NP(\Sigma_i), \qquad \Pi_{i+1} = \text{co-NP}(\Sigma_i).$$

Notice that these definitions can be relativized as well. The Σ and Π classes of the polynomial-time hierarchy can be equivalently characterized in terms of alternating polynomially length-bounded existential and universal quantifications. For such and other properties of the polynomial-time hierarchy, we refer the reader to [23, 20, 3].

Let E (NE) be the class of sets recognizable deterministically (nondeterministically) in exponential (i.e. $2^{O(n)}$) time. It has been shown [5, 7, 8] that $E \neq NE$ if and only if there exist sets over a one-letter alphabet in $NP - P$ if and only if there exist sparse sets in $NP - P$.

We assume the existence of some easy-to-compute pairing functions $\langle\ \rangle$ whose inverses are also easy to compute. For a class of sets $\mathscr{C}$ and a class of functions $\mathscr{F}$ from $\mathbb{N}$ to Σ^* let $\mathscr{C}/\mathscr{F}$ [11] be the class of sets A for which there is a set $B \in \mathscr{C}$ and a function $h \in \mathscr{F}$ such that for all $x \in \Sigma^*$,

$$x \in A \Leftrightarrow \langle x, h(n) \rangle \in B,$$

where $n = |x|$. For convenience, we write in the following $B(w)$ to denote the set $\{x \mid \langle x, w \rangle \in B\}$.

We are particularly interested in two special cases for the class $\mathscr{F}$. Let $\mathscr{F} = poly$ denote the class of polynomially bounded functions (i.e. $h \in poly$ if and only if $|h(n)| \leqslant p(n)$ for some polynomial p, depending on h.) Let $\mathscr{F} = log$ denote the class of logarithmically bounded functions (i.e. $h \in log$ if and only if $|h(n)| \leqslant c \cdot \log n$ for some constant c, depending on h). Briefly summarized, the main results of Karp and Lipton [11] concerning such nonuniform complexity classes are:

(a) If $\mathscr{C}$ is included in P/log where $\mathscr{C}$ is NP or PSPACE then $P = \mathscr{C}$.

(b) If NP is included in $P/poly$ then the polynomial hierarchy "collapses" to the class Σ_2 (i.e. $\Sigma_2 = \Pi_2 = \Sigma_3 = \Pi_3 = \cdots$).

(c) If PSPACE is included in $P/poly$ then, in addition to (b), $PSPACE = \Sigma_2$.

A further extension has been obtained by Yap [24]: If co-NP is included in $NP/poly$ then the polynomial hierarchy collapses to Σ_3. Furthermore, it has been shown that the classes $P/poly$, $NP/poly$, $PSPACE/poly$ are the same as $\{P(S) \mid S \text{ is a sparse set}\}$, $\{NP(S) \mid S \text{ is a sparse set}\}$, $\{PSPACE(S) \mid S \text{ is a sparse set}\}$, respectively.

(This is a result due to Meyer cited in [4], see also [19].) Intuitively, attaching polynomially bounded advice means the same as relativizing the underlying class with an arbitrary sparse oracle set.

Additionally, the class P/*poly* is known to be the same as the class of sets with *polynomial-size circuits* [17] (see also [4, 20] for further background) and, similarly, NP/*poly* can be characterized as the class of sets with *polynomial-size generators* [24, 19]. Several characterizations of the class PSPACE/*poly* can be found in [2]. Karp and Lipton [11] claim that P/*log* corresponds to a class of sets with small circuits that have "easy descriptions". A rigorous characterization can be found in [3, Theorem 5.8]. Here we will show that (in the proper context) logarithmic advice is the same as a sparse oracle in NP.

For technical reasons, we also consider the class strong-$\mathscr{C}/\mathscr{F}$ (cf. [13]) where $A \in$ strong-$\mathscr{C}/\mathscr{F}$ if there is a set $B \in \mathscr{C}$ and a function $h \in \mathscr{F}$ such that for all $x \in \Sigma^*$ and all $m \geqslant |x|$,

$$x \in A \Leftrightarrow \langle x, h(m) \rangle \in B.$$

It is easy to see that for any class $\mathscr{C}$ closed under $\leqslant_m^P$-reducibility (see below), strong-$\mathscr{C}/poly = \mathscr{C}/poly$. Only for advice lengths smaller than polynomial, there is a difference.

Proposition 1.1. *For every class $\mathscr{C}$ of recursive sets,* P/*log* $\not\subseteq$ strong-$\mathscr{C}/log$.

Proof. Define a set A over the one-letter alphabet $\{0\}$ such that its characteristic sequence is an infinite Kolmogorov-random string (that means that every finite initial segment of it has almost linear Kolmogorov complexity). Such a tally language A is obviously in P/*log* (even in P/1), but no initial run of A's characteristic sequence can be described by $\log n$ bits; therefore, $A \notin$ strong-$\mathscr{C}/log$ for every recursive $\mathscr{C}$. □

If we mention complete sets A for certain classes $\mathscr{C}$ then we mean polynomial-time *many-one* completeness, that is, $A \in \mathscr{C}$ and for all $B \in \mathscr{C}$, $B \leqslant_m^P A$. Another reducibility mentioned is *strong nondeterministic Turing reducibility* $\leqslant_T^{sn}$ which can be defined as $A \leqslant_T^{sn} B$ if and only if $A \in \mathrm{NP}(B) \cap \text{co-NP}(B)$. This is equivalent [21, 15] to $\mathrm{NP}(A) \subseteq \mathrm{NP}(B)$.

Relativized classes, such as P(NP) are sometimes written in the form P^{NP}. Additionally, $\mathrm{P}^{\mathrm{NP}}[log]$ means that the basis oracle machine asks only $O(\log n)$ queries to the oracle. Kadin [10] has shown that co-NP $\subseteq$ NP(S) for a sparse set $S \in$ NP implies that the polynomial-time hierarchy collapses to the class $\mathrm{P}^{\mathrm{NP}}[log]$.

The classes of the low and high hierarchy in NP were introduced in [18] and further analyzed in [14]. We give only the definition of the classes $\hat{L}_i$ for $i \geqslant 1$,

$$\hat{L}_i = \{ A \in \mathrm{NP} \mid \Delta_i(A) \subseteq \Delta_i \},$$

and refer the reader to [18, 14] for their properties.

2. Main results

We start with a theorem which says that log advice is, in a sense, equivalent to a sparse oracle in NP. Recall that poly advice indeed is equivalent to an arbitrary sparse oracle.

Theorem 2.1. (a) *If* $A \in \text{co-NP} \cap (\text{NP}/\log)$ *then* $A \in \text{NP}(S)$ *for some sparse set* $S \in \text{NP}$.
(b) *If* $A \in \text{co-NP} \cap (\text{P}/\log)$ *then* $A \in \text{P}(S)$ *for some sparse set* $S \in \text{NP}$.

Proof. To prove (a), assume that $A \in \text{co-NP} \cap (\text{NP}/\log)$. Let $B \in \text{NP}$ such that for all x, $x \in A \Leftrightarrow x \in B(w_n)$, where $n = |x|$ and w_n denotes the advice for length n; let $|w_n| \leqslant c \cdot \log n$. Consider the set

$$S = \{\langle 0^n, y\rangle \mid |y| \leqslant c \cdot \log n \wedge \exists x[|x| = n \wedge x \in B(y) \wedge x \notin A]\}.$$

Informally, S consists of those advice strings that give false "yes" answers for some string x. Since there are only polynomially many y's of logarithmic length, S is sparse. To decide S in NP, guess x and check that it is both in $B(y)$ and in $\bar{A}$. Now we claim the equality

$$A = \{x \mid \exists y(|y| \leqslant c \cdot \log |x| \wedge \langle 0^n, y\rangle \notin S \wedge x \in B(y))\}.$$

Indeed, if w_n is the correct advice for length n then $\langle 0^n, w_n\rangle \notin S$ and, thus, for $x \in A$ there is $y = w_n$ for which $\langle 0^n, y\rangle \notin S \wedge x \in B(y)$ holds. Conversely, assume $x \notin A$; if any y is found of the appropriate length for which $x \in B(y)$, then $x \in B(y) - A$; therefore, $\langle 0^n, y\rangle \in S$ and x is not in the set on the right-hand side. The form of this characterization of A proves that $A \in \text{NP}(S)$.

Finally, to prove (b), observe that under the hypothesis that $B \in \text{P}$, the characterization of A becomes $\text{P}(S)$, since the quantification is logarithmically bounded.

The reader should notice the unusual "one-sided correctness" that the sparse oracle S above has. This one-sided correctness is the reason for the membership of S in NP and, on the other hand, it is in this context still sufficiently strong to allow the basis algorithm to obtain "full correctness" by asking several oracle queries. Also notice that the set A is in a slightly smaller class than $\text{NP}(S)$ since the number of points in the entire $\text{NP}(S)$ computation tree at which a query is asked, is polynomial.

Note that S can be taken as well in co-NP by defining it in terms of the "opposite kind of mistakes": form S' with all pairs $\langle 0^n, y\rangle$, where $|y| = c \cdot \log n$, and such that every x of length n in $B(y)$ is also in A. This can be checked in co-NP, and now A is defined by

$$A = \{x \mid \exists y(|y| = c \cdot \log |x| \wedge \langle 0^n, y\rangle \in S \wedge x \in B(y))\}.$$

The argument is essentially the same. Also, by simple complementation, we can obtain that if $A \in \text{NP} \cap (\text{co-NP}/\log)$ then $A \in \text{co-NP}(S)$, where $S \in \text{NP}$.

We use this Theorem 2.1 and the above observations to obtain the following equivalences.

Corollary 2.2. *The following are equivalent*:
(a) co-NP$\subseteq$NP/*log*.
(b) co-NP$\subseteq$NP(S) *for some sparse* $S\in$NP.
(c) *There is a sparse* $\leqslant_T^{sn}$*-complete set for* NP.

Proof. (a)$\Rightarrow$(b). Let TAUT be the set of boolean tautologies, which is $\leqslant_m^p$-complete for co-NP. By (a), TAUT$\in$NP/*log*. Thus, Theorem 2.1 yields TAUT$\in$NP(S) for a sparse set S in NP. Since NP(S) is closed under $\leqslant_m^p$-reducibility, co-NP$\subseteq$NP(S).

(b)$\Rightarrow$(c). Assume co-NP$\subseteq$NP(S) for sparse $S\in$NP. We show that S is $\leqslant_T^{sn}$-hard for NP. Since co-NP$\subseteq$co-NP(S) trivially, co-NP$\subseteq$NP(S)$\cap$co-NP(S) and, by complementation, NP$\subseteq$NP(S)$\cap$co-NP(S). Thus, $A\leqslant_T^{sn}S$ for every set $A\in$NP.

(c)$\Rightarrow$(a). By (c) and complementation, co-NP$\subseteq$NP(S)$\cap$co-NP(S)$\subseteq$NP(S). The advice will consist of the census of S up to length $p(n)$ for a polynomial p; then nondeterminism is used to find out all of S up to $p(n)$, obtaining an NP algorithm without an oracle. Formally, let $A\in$NP(S) via a nondeterministic oracle machine M whose queries on inputs of length n are of length at most $p(n)$. On input $\langle x, k\rangle$, where $|x|=n$ and k is the census of S up to length $p(n)$, the following nondeterministic algorithm accepts A:

INPUT $\langle x, k\rangle$
GUESS a set S' of k different strings of length at most $p(n)$
FOR each string w in S' DO
  nondeterministically check that $w\in S$
CHECK that $x\in L(M, S')$ and is so, halt accepting.

Since k can be written down in $O(\log n)$ bits, we obtain that $A\in$NP/*log*. □

The class NP/*log* is closed under polynomially length-bounded existential quantification. Therefore, using the quantifier characterization of the polynomial-time hierarchy, all of the following statements are easily seen to be equivalent to co-NP$\subseteq$NP/*log*:

(a) There is a sparse $\leqslant_T^{sn}$-complete set for co-NP.
(b) co-NP$\subseteq$NP(S) for some sparse $S\in$co-NP.
(c) $\Sigma_2\subseteq$NP(S) for some sparse $S\in$NP.
(d) $\Sigma_2\subseteq$NP(S) for some sparse $S\in$co-NP.
(e) $\Sigma_2\subseteq$NP/*log*.

Notice that all the classes mentioned in these facts can be complemented to obtain new facts, such as NP$\subseteq$co-NP/*log*, $\Pi_2\subseteq$co-NP/*log*, and the like. Iterating the argument and using the fact that (NP/*log*)/*log*=NP/*log*, we also obtain that co-NP$\subseteq$NP/*log* if and only if PH$\subseteq$NP/*log*. A second consequence of the theorem is the inclusion of NP$\cap$(co-NP/*log*) in the low hierarchy [19, 14]. We need the following easy lemma.

Lemma 2.3. *The classes $\hat{L}_i$ for $i \geq 2$, are closed under $\leq_T^{sm}$.*

Proof. Assume $B \in \hat{L}_i$ with $i \geq 2$. It is known [15, 21] that $A \leq_T^{sm} B$ if and only if $NP(A) \subseteq NP(B)$. Then $\Delta_i(A) = \Delta_{i-1}(NP(A)) \subseteq \Delta_{i-1}(NP(B)) = \Delta_i(B) = \Delta_i$ and, therefore, $A \in \hat{L}_i$. □

Using this fact, and the fact that sparse sets in NP are in $\hat{L}_2$ [14], we obtain Corollary 2.4.

Corollary 2.4. $NP \cap (\text{co-NP}/log) \subseteq \hat{L}_2$.

Proof. Let $A \in NP \cap (\text{co-NP}/log)$. By the remark preceding Corollary 2.2, $A \in \text{co-NP}(S)$, where $S \in NP$. Since $A \in NP$, we can state that $A \in NP(S) \cap \text{co-NP}(S)$, i.e. $A \leq_T^{sm} S \in \hat{L}_2$ [14]. By Lemma 2.3, $A \in \hat{L}_2$.

Thus, if $NP \subseteq \text{co-NP}/log$ then the polynomial-time hierarchy collapses to Δ_2. It should be observed that this collapse can be improved.

Corollary 2.5. *If* $\text{co-NP} \subseteq NP/log$ *then the polynomial-time hierarchy collapses to* $P^{NP}[log]$.

Proof. In [10] it is shown that if $\text{co-NP} \subseteq NP(S)$ for some sparse $S \in NP$ then the indicated collapse holds. Corollary 2.2 asserts that $\text{co-NP} \subseteq NP(S)$ for some sparse $S \in NP$ is equivalent to $NP \subseteq \text{co-NP}/log$.

Next, we show that questions about exponential-time classes can be connected with questions about log-advice classes.

Theorem 2.6. (a) $(NP \cap (P/log)) - P \neq \emptyset$ *if and only if* $E \neq NE$.
(b) $(\text{co-NP} \cap (NP/log)) - NP \neq \emptyset$ *if and only if* $NE \neq \text{co-NE}$.

Proof. Right to left implications are immediate from the fact that all tally sets are in P/log, and the fact that $E \neq NE$ implies there are tally sets in $NP - P$ (similar for $NE \neq \text{co-NE}$ and sparse sets in $NP - \text{co-NP}$).

Conversely, assume $E = NE$. By [7, 8], no sparse sets exist in $NP - P$. By Theorem 2.1, using complementation, we know that the sets in $NP \cap (P/log)$ are in $P(S)$ for a sparse set $S \in NP$ and, therefore, $S \in P$. Thus, $NP \cap (P/log) = P$. Similarly, if NE is closed under complementation then all sparse sets in NP are also in co-NP [7, 8, 9]. Thus, Theorem 2.1 yields that sets in $\text{co-NP} \cap (NP/log)$ are in $NP(S)$, where $S \in NP \cap \text{co-NP}$ and, therefore, $\text{co-NP} \cap (NP/log) \subseteq NP$. □

It is also interesting to note that $NP \cap (P/log)$ has $\leq_T^p$-complete sets, namely, those tally sets that are $\leq_T^p$-complete for the class of sparse sets in NP (see [7, 8, 9]).

Proposition 2.7. (cf. Hartmanis [7]). *Let* $T_0 = \text{tally}(K)$, *where* K *is any* $\leqslant_m^p$*-complete set for* NE. *Then* T_0 *is* $\leqslant_T^p$*-complete for* $\text{NP} \cap (\text{P}/log)$.

Proof. It is known [7, 8] that T_0 is $\leqslant_T^p$-complete for the class of sparse sets in NP. Of course, $T_0 \in \text{NP} \cap (\text{P}/log)$. Given $A \in \text{NP} \cap (\text{P}/log)$, we see from Theorem 2.6 that $A \in \text{P}(S)$ for a sparse $S \in \text{NP}$ and, therefore, $S \in \text{P}(T_0)$. Thus, $A \in \text{P}(T_0)$. □

Further, we want to point out the incomparability of NP/*log* and P/*poly* under reasonable assumptions.

Theorem 2.8. (a) *If* $\Sigma_2 \neq \Pi_2$ *then* $\text{NP}/log \not\subseteq \text{P}/poly$ [11].
(b) $\text{P}/poly \not\subseteq \text{NP}/log$.

Proof. Statement (a) follows from [11] since $\text{NP} \subseteq \text{NP}/log$, and $\text{NP} \subseteq \text{P}/poly$ implies $\Sigma_2 = \Pi_2$. Statement (b) can be proved by considering a sparse set S whose characteristic function at each length n consists of a string of length n of high (unbounded) Kolmogorov complexity, followed by $2^n - n$ zeros. Such a set cannot be decided with $O(\log n)$ advice in a recursive manner, since, otherwise, the advice plus a constant length program would allow to recover the first n bits of the characteristic function of S, contradicting its high Kolmogorov complexity. □

Note that Theorem 2.8(b) can be generalized, similar to Proposition 1.1: *For every class* $\mathcal{C}$ *of recursive (even r.e. or arithmetical) sets,* $\text{P}/poly \not\subseteq \mathcal{C}/log$.

Several of Karp and Lipton's proofs [11] take advantage of the self-reducibility structure of typical $\mathcal{C}$-complete sets, where $\mathcal{C}$ is P, NP or PSPACE. The following definition of self-reducibility refers to the length ordering of strings. More general definitions can be found in [12, 16].

Definition 2.9. A set A is *self-reducible* if there is a deterministic, polynomial-time oracle machine M such that $A = L(M, A)$, and for each input x, M queries the oracle only for strings y, $|y| < |x|$.

Theorem 2.10. *If* A *is a self-reducible set with* $A \in \text{strong-P}/log$, *then* $A \in \text{P}$.

Proof. Let M be the self-reducing machine for A according to Definition 2.9.

Since $A \in \text{strong-P}/log$, there is a set $B \in \text{P}$ together with a sequence $\{w_n\}_{n \in \mathbb{N}}$ of strings such that $|w_n| \leqslant c \cdot \log n$ for some constant c and all n, and $A_n = B(w_m)_n$ for all n and $m \geqslant n$.

Let a polynomial-time computable 2-placed predicate, *consistent*, be defined as

$$\text{consistent}(x, w) \text{ iff } (x \in B(w) \Leftrightarrow x \in L(M, B(w))).$$

Intuitively, consistent (x, w) is true if the "advice" w together with the "advice interpreter" B are consistent with the first level of the self-reduction structure of x induced by M. Notice that consistent (x, w_m) is true for all $m \geqslant |x|$.

Now, the following algorithm will be shown to recognize A in polynomial time; hence, $A \in \mathrm{P}$.

```
INPUT x; {|x| = n};
FOR u, |u| ≤ c·log n DO
BEGIN
  b := TRUE;
  FOR v, |v| ≤ c·log n DO b := b AND test (x, u, v);
  IF b THEN
    IF x ∈ L(M, B(u))
      THEN halt accepting
      ELSE halt rejecting;
END
```

Here, the recursive procedure test (x, u, v) operates as follows.

```
PROCEDURE test (x, u, v): BOOLEAN;
BEGIN
  IF (x ∈ L(M, B(u)) ⇔ x ∈ L(M, B(v)))
    THEN RETURN TRUE
    ELSE
    BEGIN
        y := the first query string in the computations of M^B(u)
             and M^B(v) on input x, where the oracles B(u) and
             B(v) disagree in their respective answers;
        IF NOT consistent (y, u) THEN RETURN FALSE;
        IF NOT consistent (y, v) THEN RETURN TRUE;
        RETURN test (y, u, v);
    END
END
```

The idea here is similar to an argument in [11]. Every potential advice u plays a "game" (in the notation of Karp and Lipton) against every possible other advice v. Every good advice u will win all games against v, and every bad advice u will lose a game against some v.

By polynomial well-foundedness of the self-reduction structure of A, it follows that the (tail) recursion of test is polynomially bounded in depth. Whenever a string in the self-reducibility structure of M on x is reached where M does not query its oracle, then $x \in L(M, B(u)) \Leftrightarrow x \in L(M, B(v))$ is true, and the recursion ends. (Observe that the

recursion may end before reaching such a string.) Therefore, the procedure test runs in polynomial time (in $|x|+|u|+|v|$). The main program is, therefore, polynomial-time in $|x|$.

Regarding the correctness, we prove two claims.

Claim 1. *For every string* x ($|x|=n$), *every* $m \geqslant n$, *and every* v ($|v| \leqslant c \cdot \log n$), test (x, w_m, v) *evaluates to* TRUE.

Proof. Suppose the contrary. Choose x minimal such that for some m and some v, test (x, w_m, v) = FALSE. This string x cannot be a "leaf" in the self-reducibility structure of M on x, otherwise the procedure test would return TRUE in the first line. That is, M on input x has to have at least one oracle query, and the ELSE branch is entered. Since the oracles $B(w_m)$ and $B(v)$ disagree on the status of x, there must be a first oracle query of a string y, on which the oracles disagree. The test consistent (y, w_m) is true by definition of w_m. Also, the next test for consistent (y, v) is true since test is assumed to return FALSE. Therefore, the control reaches the recursive call of test (y, w_m, v) which, by assumption, evaluates to FALSE. But this is a contradiction to the minimality of the choice of x. □

Claim 2. *For every string* x ($|x|=n$), *every* u ($|u| \leqslant c \cdot \log n$), *and every* $m \geqslant n$, *if* test (x, u, w_m) *evaluates to* TRUE, *then*

$$x \in L(M, B(u)) \Leftrightarrow x \in L(M, B(w_m)).$$

(*Therefore*, $x \in L(M, B(u)) \Leftrightarrow x \in A$.)

Proof. Assume the contrary. Fix a string u for which the predicate fails, and choose x ($|x|=n$) minimal such that for some m, test (x, u, w_m) = TRUE, but

$$x \in L(M, B(u)) \Leftrightarrow x \notin L(M, B(w_m)).$$

Consider the execution of the procedure test (x, u, w_m). By the above assumption, the ELSE branch is entered. Therefore, M on input x has at least one oracle query, and a query string y can be determined such that

$$y \in B(u) \Leftrightarrow y \notin B(w_m). \tag{1}$$

By the assumption that test (x, u, w_m) = TRUE, it must be the case that consistent (y, u) is true, i.e.

$$y \in L(M, B(u)) \Leftrightarrow y \in B(u). \tag{2}$$

Also, by definition of w_m, consistent (y, w_m) is true, i.e.

$$y \in L(M, B(w_m)) \Leftrightarrow y \in B(w_m). \tag{3}$$

Therefore, the control reaches the recursive call of test (y, u, w_m). By the assumption, this call returns TRUE. By minimality of x, and since $y < x$, the assertion of the claim holds for y, i.e.

$$y \in L(M, B(u)) \Leftrightarrow y \in L(M, B(w_m)). \tag{4}$$

Now, combining the statements (1), (2), (3) and (4) gives a contradiction, which proves the claim. □

Proof of Theorem 2.10 (*conclusion*). The correctness of the algorithm follows from Claim 1 and 2 as follows. Whenever the outer FOR-loop in the main program finds some u with test $(x, u, v) = \text{TRUE}$ for all v (especially for $v = w_n$), then, by Claim 2, the decision of the algorithm for accepting or rejecting is correct.

On the other hand, Claim 1 guarantees that at least one such u, e.g. $u = w_n$, will be found in the outer FOR-loop.

This completes the proof of the Theorem.

For simplicity, the above proof was given w.r.t. self-reducibility defined on the length order. It is easy to see that this is not essential. The same proof goes through when using the more general definitions from [12, 16] where just a polynomially well-founded order, not necessarily length-respecting, on the "self-reduction tree" is required.

Furthermore, if the self-reduction tree can, by padding properties of the considered language A, be assumed to consist only of strings of the same length as the input string x, then the assumption "$A \in$ strong-P/*log*" can be weakened to "$A \in$ P/*log*". Since all "natural" complete sets for classes $\mathcal{C}$ in PSPACE do have such padding properties (provided $\mathcal{C}$ has complete sets), we immediately obtain (for the cases $\mathcal{C} = \text{NP}$ and $\mathcal{C} = \text{PSPACE}$) the following corollaries.

Corollary 2.11 (Karp and Lipton [11]). *If* $\text{NP} \subseteq \text{P/log}$ *then* $\text{NP} = \text{P}$.

Corollary 2.12 (Karp and Lipton [11]). *If* $\text{PSPACE} \subseteq \text{P/log}$ *then* $\text{PSPACE} = \text{P}$.

Proof. Use the fact that the set of valid quantified Boolean formulas, *QBF*, has a very simple (with appropriate encoding, even length-respecting) self-reducibility structure, and that *QBF* is PSPACE-complete. □

Actually, this simple (namely, 2-truth-table and positive) self-reducibility structure of *QBF* was used by Karp and Lipton in their original proof of this corollary. Our theorem extends their "round-robin tournament" method to the more general situation of an adaptive self-reduction. Karp and Lipton's original argument uses positiveness of the reduction and is, therefore, not directly applicable to this more general

situation. For example, the language

$$\#SAT = \{\langle F, k\rangle \mid \text{the formula } F \text{ has at least } k \text{ satisfying assignments}\},$$

which is complete for the class PP (see [22, 6]), has a more general (adaptive) self-reducibility structure that uses binary search (see [1, 20]). The following application of Theorem 2.10 is, therefore, new.

Corollary 2.13. *If* $\mathrm{PP} \subseteq \mathrm{P}/log$ *then* $\mathrm{PP} = \mathrm{P}$.

It follows immediately from Theorem 2.10 since the padding properties of $\#SAT$ imply that it is in P/log if and only if it is in strong-P/log.

Acknowledgment

The authors thank Rainer Schuler for pointing out an error in an earlier version of this paper.

Note added in proof

Further results regarding P/log and circuits have been obtained by Hermo and Mayordormo (Report LSI-91-20, Univ. Politècnica de Catalunya, Barcelona, Spain).

References

[1] J.L. Balcázar, R.V. Book and U. Schöning, The polynomial-time hierarchy and sparse oracles, *J. Assoc. Comput. Mach.* **33** (1986) 603–617.

[2] J.L. Balcázar, J. Diaz and J. Gabarró, On characterizations of the class PSPACE/poly, *Theoret. Comput. Sci.* **52** (1987) 1–17.

[3] J.L. Balcázar, J. Diaz and J. Gabarró, *Structural Complexity I*, EATCS Monographs Vol. 11 (Springer, Berlin, 1988).

[4] L. Berman and J. Hartmanis, On isomorphism and density of NP and other complete sets, *SIAM J. Comput.* **6** (1977) 305–327.

[5] R.V. Book, Tally languages and complexity classes, *Inform. and Control* **26** (1974) 186–193.

[6] J. Gill, Computational complexity of probabilistic complexity classes, *SIAM J. Comput.* **6** (1977) 675–695.

[7] J. Hartmanis, On sparse sets in NP – P, *Inform. Process. Lett.* **16** (1983) 55–60.

[8] J. Hartmanis, N. Immerman and V. Sewelson, Sparse sets in NP – P: EXPTIME versus NEXPTIME, in: *Proc. 15th ACM Sympos. Theory of Comput.* (1983) 382–391.

[9] J. Hartmanis and Y. Yesha, Computation times of NP sets of different densities, *Theoret. Comput. Sci.* **34** (1984) 17–32.

[10] J. Kadin, $\mathrm{P}^{\mathrm{NP}[\log n]}$ and sparse Turing-complete sets for NP, in: *Proc. 2nd Conf. Structure in Complexity Theory*, (IEEE Computer Soc., Silver Spring, MD, 1987) 33–40.

[11] R.M. Karp and R.J. Lipton, Some connections between nonuniform and uniform complexity classes, in: *Proc. 12th ACM Symp. Theory of Comput. Sci.* (1980) 302–309.

[12] K. Ko, On self-reducibility and weak P-selectivity, *J. Comput. System Sci.* **26** (1983) 209–221.
[13] K. Ko, On helping by robust oracle machines, *Theoret. Comput. Sci.* **52** (1987) 15–36.
[14] K. Ko and U. Schöning, On circuit-size complexity and the low hierarchy in NP, *SIAM J. Comput.* **14** (1985) 41–51.
[15] T.J. Long, Strong nondeterministic polynomial-time reducibilities, *Theoret. Comput. Sci.* **21** (1982) 1–25.
[16] A. Meyer and M. Paterson, With what frequency are apparently intractable problems difficult? MIT Tech. Report No. 126, 1979.
[17] N. Pippenger, On simultaneous resource bounds, in: *Proc. 20th IEEE Sympos. Foundations of Computer Science* (1979) 307–311.
[18] U. Schöning, A low and a high hierarchy within NP, *J. Comput. System Sci.* **27** (1983) 14–28.
[19] U. Schöning, On small generators, *Theoret. Comput. Sci.* **34** (1984) 337–341.
[20] U. Schöning, *Complexity and Structure*, Lecture Notes in Computer Science, Vol. 211 (Springer, Berlin, 1985).
[21] A.L. Selman, Polynomial-time enumeration reducibility, *SIAM J. Comput.* **7** (1978) 440–447.
[22] J. Simon, On the difference between one and many, in: *Proc. Internat. Conf. Automata, Languages and Programming*, Lecture Notes in Computer Science Vol. 52 (Springer, Berlin, 1977) 480–491.
[23] L.J. Stockmeyer, The polynomial-time hierarchy, *Theoret. Comput. Sci.* **3** (1977) 1–22.
[24] C.K. Yap, Some consequences of non-uniform conditions on uniform classes, *Theoret. Comput. Sci.* **26** (1983) 287–300.